DATS 6313_10: Time series Analysis and Modelling
Dr. Reza Jafari
Final Term Project
Atharva Haldankar
George Washington University
05-03-2022

INDEX		
Chap.	**Topic**	**Pg. No.**
1	Introduction	5
2	Description of the Data i) pre-processing ii) Dependent variable vs Time iii) ACF/PACF plot iv) Correlation Matrix with Heat Map v) Train-Test Split	5-8
3	Stationarity	9-10
4	Time series decomposition	10-11
5	Holt Winters Method	11-12
6	Feature Selection	12-13
7	Base Models i) Average Method ii) Naïve method iii) Drift Method iv) SES method	14-17
8	Multiple Linear Regression (MLR)	19
9	ARMA, ARIMA & SARIMA model order selection.	20
10	Levenberg Marquardt Algorithm	21-23
11	Final Model Selection	24
12	h-step Prediction	25
13	Summary and conclusions	26
14	References	26
15	Appendix	27-36

No.	Figures	Pg. No.
1	Plot for electricity usage every 60 minutes VS time	7
2	Change in the unit cost every month	8
3	ACF & PACF plot	8
4	Heat map for correlation amongst continuous variables.	8
5	ACF & PACF Plot for hourly electricity usage.	9
6	Rolling mean and rolling Variance	9
7	ADF and KPSS test for the data	9
8	STL decomposition	10
9	Plot for seasonality, trend and residual	10
10	Original Data vs Seasonally adjusted data.	10
11	Original data vs Detrended data	11
12	ACF for prediction error (Holt Winters)	11
13	Holt Winters method	12
14	Average method Forecasting	14
15	Naive method forecasting	15
16	Drift method forecasting	16
17	SES method forecasting	17
18	GPAC	19
19	ACF PACF for data after differencing	19
20	ACF for residuals (ARMA)	20
21	Train and prediction for ARMA(2,0)	20
22	ACF for ARIMA	21
23	ARIMA Train vs Prediction.	22
24	SARIMA train vs prediction	23
25	25 h-step prediction for MLR	24

ABSTRACT

This report focuses on developing a python code. The objective of this report is to apply the course learning objectives to a real dataset for modeling & prediction. A real world data is acquired and with the help of all the tools and knowledge from the learnt from this course is applied to find the best model to forecast.

1: Introduction

Time Series Analysis was performed on a dataset which records the electricity usage every 15 minutes from Kaggle. The data was then transformed to hourly electricity usage. A time series decomposition was done on this data and plotted the seasonally adjusted data and the detrended data.
After the time series decomposition, different models are used to check the accuracy of each model and which model works the best for this data. The final model is then selected and the h-step prediction is then performed.

2 : Description of the dataset

Pre-processing of the data

The dataset is taken from Kaggle. First, the end time and the date column were combined and was set as the index.

After setting the index, a column named “season” was added, which indicates the season. This is a categorical column which consists of 4 categories (1,2,3,4) which indicates the 4 seasons of the year.
1: Spring
2: Summer
3: Fall
4: Winter

Another column named “HOLIDAY” was added, which indicates whether it was federal holiday or not. This column too is a categorical column which consists of two categories (0,1)
0: Not a Holiday
1: Holiday

The “COST” column records the cost of the electricity used. The cost was recorded as a string which was later changed to float.

```
df  = pd.read_csv("/Users/atharvah/GWU/Untitled/personal/Time_Series/Labs/D202.csv")
df.DATE = pd.to_datetime(df.DATE + ' ' + df["END TIME"])
df = df.drop(columns=["START TIME","END TIME"])
df = df.set_index(df.DATE)
# 1: spring,2: summer,3: fall,4: winter
df["season"]= df.index.month%12 // 3 + 1
Run Cell | Run Above | Debug Cell | Go to [456]
#%%
# Citation: JAGANADH GOPINADHAN
# Check if holiday or not
cal = calendar()
holidays = cal.holidays(start = df.index.min(), end = df.index.max())
df["HOLIDAY"] = df.index.strftime('%Y-%m-%d').isin(holidays.strftime('%Y-%m-%d'))
df["HOLIDAY"] = pd.Categorical(df["HOLIDAY"])
df.season = pd.Categorical(df.season)
Run Cell | Run Above | Debug Cell | Go to [457]
#%%
df.COST = df.COST.str[1:]
df.COST = df.COST.astype(float)
```

Columns were then inspected for NA values. It was discovered that the “NOTES” columns only consisted of NA values while no other columns had any NA values.

```
display(df.head())
display(df.tail())
missing_vals = df.isna().sum()
print("The numeber of Missing Values:", missing_vals)
```

```
The numeber of Missing Values: TYPE
DATE           0
USAGE          0
UNITS          0
COST           0
NOTES      70368
season         0
HOLIDAY        0
dtype: int64
```

To have a better picture of the scenario, 3 new data frames were created which are derived from the original dataset.

hourly_usage_df: this records the electricity usage for every hour. This data frame is used for this project.

daily_usage_df: this records the electricity usage for 24 hours.

monthly_usage_df: this records the electricity usage monthly.

```
#%%
# Citation : SANKET SHARMA
# Hourly
hourly_usage_df = df.resample("1H", on="DATE").sum()
display(hourly_usage_df.head())

# Daily
daily_usage_df = hourly_usage_df.resample("1D").sum()
display(daily_usage_df.head())

# Monthly
monthly_usage_df = daily_usage_df.resample("1M").sum()
display(monthly_usage_df.head())
```

The same columns were added to these data frames along with “per_unit_cost” and “usage_month”. These columns record the rate for electricity and the month of usage respectively.

```
#%%
monthly_usage_df["per_unit_cost"] = monthly_usage_df["COST"]/monthly_usage_df["USAGE"]
monthly_usage_df["year"] = monthly_usage_df.index.year.astype(str)
monthly_usage_df["month_name"] = monthly_usage_df.index.month_name().str[:3]

monthly_usage_df["usage_month"] = monthly_usage_df["month_name"] + "-" + monthly_usage_df
["year"]

monthly_usage_df = monthly_usage_df.drop(columns=["year", "month_name"], errors="ignore")

hourly_usage_df["year"] = hourly_usage_df.index.year.astype(str)
hourly_usage_df["month_name"] = hourly_usage_df.index.month_name().str[:3]

hourly_usage_df["usage_month"] = hourly_usage_df["month_name"] + "-" + hourly_usage_df
["year"]
hourly_usage_df["season"]= hourly_usage_df.index.month%12 // 3 + 1
hourly_usage_df.season = pd.Categorical(hourly_usage_df.season)

hourly_usage_df = hourly_usage_df.drop(columns=["year", "month_name"], errors="ignore")
hourly_usage_df["HOLIDAY"] = hourly_usage_df.index.strftime('%Y-%m-%d').isin(holidays.
strftime('%Y-%m-%d')).astype(int)
hourly_usage_df["HOLIDAY"] = pd.Categorical(hourly_usage_df["HOLIDAY"])

display(hourly_usage_df.head())
```

Dependent variable and it’s plot with time

The dependent variable for this project is the electricity usage for every 1 hour.
The independent variables are season, holiday and cost.

The electricity usage for every 60 minutes is first plotted against time.
It can be clearly seen that this data data is additive.

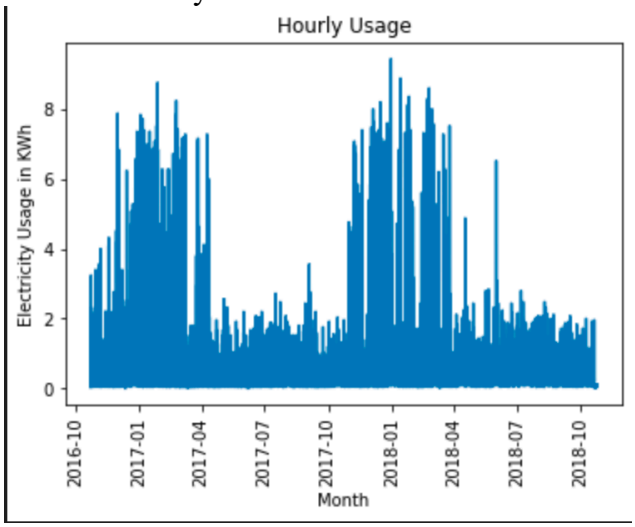

Figure 1: Plot for electricity usage every 60 minutes VS time

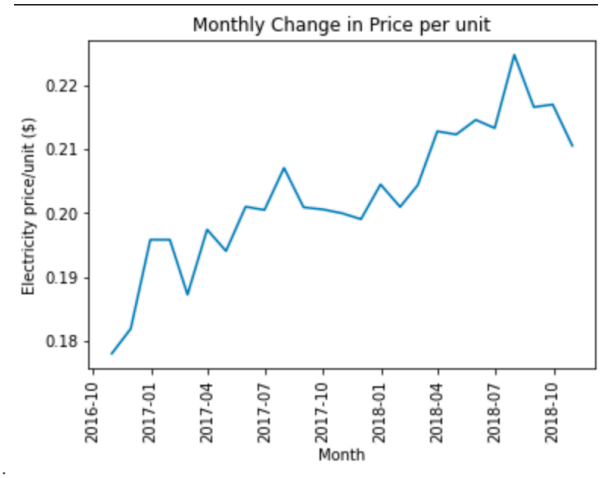

Figure 2 Change in the unit cost every month

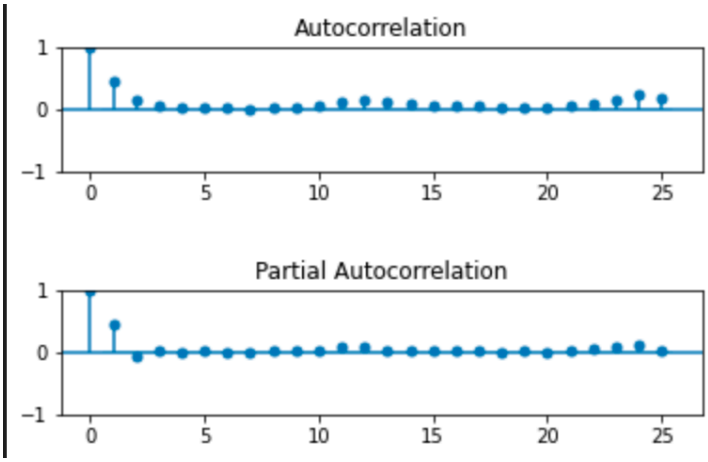

Figure 3 ACF & PACF plot

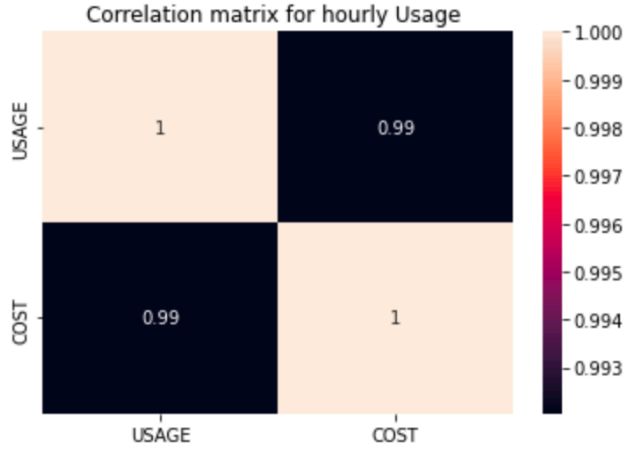

Figure 4 Heat map for correlation amongst continuous variables.

The hour_usage_df is then split for training and testing.
The training data consists of 80% and testing consists of 20%.

```
X_train, X_test = train_test_split(hourly_usage_df,test_size=0.2,shuffle=False)
```

3 : Stationarity

From the figure below it can be seen that there is an immediate cutoff after lag 1 for both ACF & PACF plot indicating that the data is stationary.

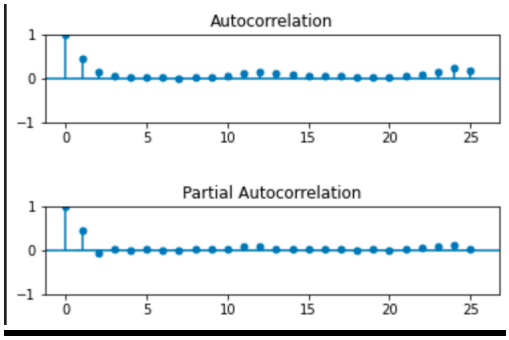

Figure 5 ACF & PACF Plot for hourly electricity usage.

The rolling mean and the rolling variance converges into a straight line which indicates that the data is stationary.

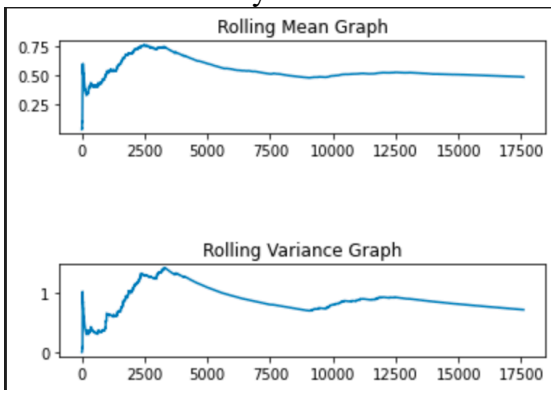

Figure 6 Rolling mean and rolling Variance

```
✓ tx.ADF_Cal(hourly_usage_df.USAGE) ···

ADF Statistic: -12.214292
p-value: 0.000000
Critical Values:
        1%: -3.431
        5%: -2.862
        10%: -2.567


✓ tx.kpss_test(hourly_usage_df.USAGE) ···

Results of KPSS Test:
Test Statistic           3.559046
p-value                  0.010000
LagsUsed                30.000000
Critical Value (10%)     0.347000
dtype: float64
Test Statistic           3.559046
p-value                  0.010000
LagsUsed                30.000000
Critical Value (10%)     0.347000
Critical Value (5%)      0.463000
dtype: float64
```

Figure 7 ADF and KPSS test for the data

The ADF and the KPSS test confirms that the data is stationary. The p value for ADF test is less than 0.05 which allows us to reject the null hypothesis and incorporate the alternate hypothesis which means the data is stationary. For the KPSS test the opposite is observed. Hence the null hypothesis cannot be rejected and the hence the data is stationary.

4 : Time Series Decomposition

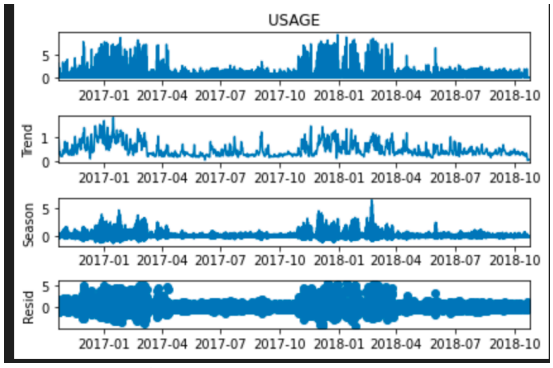

Figure 8 STL decomposition

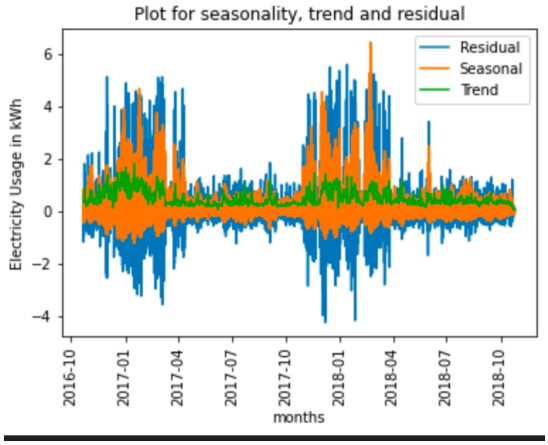

Figure 9 Plot for seasonality, trend and residual

Since the data is additive, for seasonally adjusted data, the seasonality is subtracted from the data and for detrended data, the trend is subtracted from the data as well.

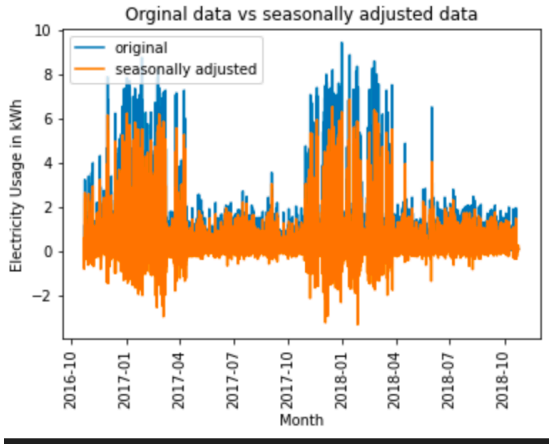

Figure 10 Original Data vs Seasonally adjusted data.

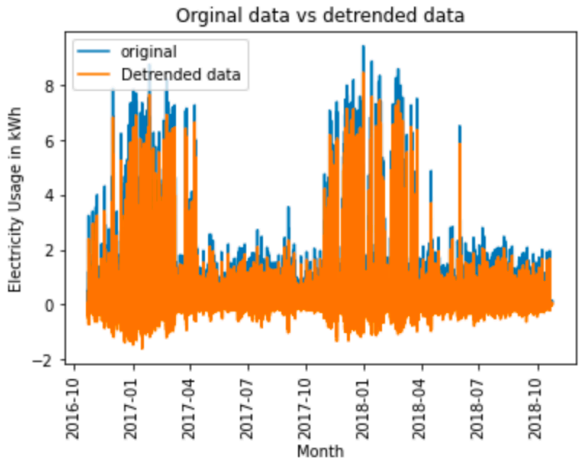

Figure 11 Original data vs Detrended data

From the strength for seasonality and strength for trend, it can be observed that the data is seasonal.

```
The strength of seasonality for this data set is:0.4592742640480071
The strength of trend for this data set is:0.2174558497078608
```

5 : Holt-Winter Method

The Mean Squared Error (MSE) for the prediction and the Mean Squared Error (MSE) for the forecast is highlighted for the holt-winter's method.

```
Name of test:Holt's Winter Method,
 'Q value':[1739.94534912   15.62528482    7.36474871   21.42361737   23.86227835],MSE for prediction
errors: 0.7407944700874283,MSE for forecast errors:0.16084638698650472,Mean of prediction
errors:nan,Variance of prediction errors:0.7407944673311574,Variance of forecast errors:
0.16082038707233487, Correlation Coeff:[[1.         0.65333688]
 [0.65333688 1.        ]]
```

Form the ACF plot for prediction error it can be seen that the error is white.

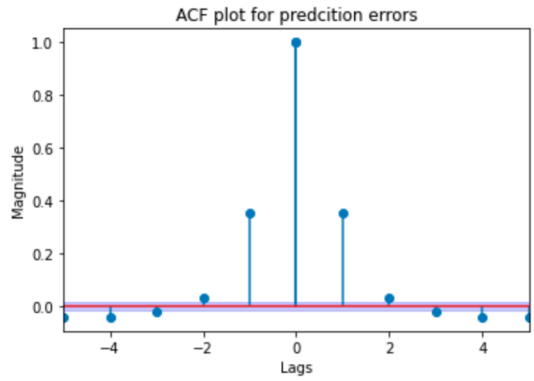

Figure 12 ACF for prediction error (Holt Winters)

Form plot it can be seen that the forecast is not very accurate.

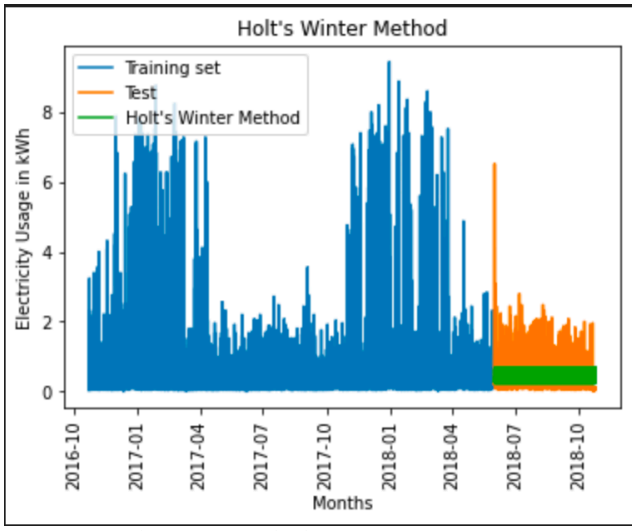

Figure 13 Holt Winters method

6 : Feature Selection

```
Singular Values = [103860.2165557    2496.1106886     461.95867446    391.91628125]
The conditional number = 16.279008843400597
```

All the singular values are greater than zero which does not indicate any collinearity. The Conditional number is also very small which indicates not very strong collinearity.

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  USAGE   R-squared:                       0.985
Model:                            OLS   Adj. R-squared:                  0.985
Method:                 Least Squares   F-statistic:                 3.127e+05
Date:                Tue, 03 May 2022   Prob (F-statistic):               0.00
Time:                        03:07:32   Log-Likelihood:                 10926.
No. Observations:               14073   AIC:                         -2.184e+04
Df Residuals:                   14069   BIC:                         -2.181e+04
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0075      0.002      3.279      0.001       0.003       0.012
COST           4.9139      0.005    958.103      0.000       4.904       4.924
season         0.0006      0.001      0.681      0.496      -0.001       0.002
HOLIDAY        0.0301      0.006      5.384      0.000       0.019       0.041
==============================================================================
Omnibus:                    13695.835   Durbin-Watson:                   1.268
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          4535143.057
Skew:                          -4.061   Prob(JB):                         0.00
Kurtosis:                      90.568   Cond. No.                         16.3
==============================================================================
```

From the results of the above model, it can be seen that the only season has p value greater than 0.05. Hence the feature needs to be dropped.

After dropping the feature “season”, there is an improvement in the conditional number. Since all the p values are less than 0.05, call the features are not zero.

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  USAGE   R-squared:                       0.985
Model:                            OLS   Adj. R-squared:                  0.985
Method:                 Least Squares   F-statistic:                 4.691e+05
Date:                Tue, 03 May 2022   Prob (F-statistic):               0.00
Time:                        03:08:28   Log-Likelihood:                 10926.
No. Observations:               14073   AIC:                         -2.185e+04
Df Residuals:                   14070   BIC:                         -2.182e+04
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0088      0.001      8.164      0.000       0.007       0.011
COST           4.9135      0.005    967.510      0.000       4.904       4.923
HOLIDAY        0.0302      0.006      5.388      0.000       0.019       0.041
==============================================================================
Omnibus:                    13698.221   Durbin-Watson:                   1.268
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          4533127.117
Skew:                          -4.063   Prob(JB):                         0.00
Kurtosis:                      90.548   Cond. No.                         6.02
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
Singular Values = [14238.18398816   479.67838847   392.33982337]
The conditional number = 6.024154536629359
```

The final features selected for selected for this model are cost and holiday.

7 : Base Models

Average method

For train set

Y(t)	Forecast	Errors	Squared Errors
0.04			
0.04	0.04	0	0
0.08	0.04	0.04	0.0016
0.04	0.0533333	-0.0133333	0.000177778
0.04	0.05	-0.01	0.0001
0.04	0.048	-0.008	6.4e-05
0.04	0.0466667	-0.00666667	4.44444e-05
0.04	0.0457143	-0.00571429	3.26531e-05
0.08	0.045	0.035	0.001225
0.04	0.0488889	-0.00888889	7.90123e-05
0.08	0.048	0.032	0.001024
0.28	0.0509091	0.229091	0.0524826
0.52	0.07	0.45	0.2025
0.04	0.104615	-0.0646154	0.00417515
0.24	0.1	0.14	0.0196

The mean squared error(MSE): 0.8391905221868133
The variance for prediction errors: 0.837231736589893

For test set

Y(t)	Forecast	Errors	Squared Errors
0.28	0.513127	-0.233127	0.0543483
0.24	0.513127	-0.273127	0.0745985
0.32	0.513127	-0.193127	0.0372981
0.28	0.513127	-0.233127	0.0543483
0.28	0.513127	-0.233127	0.0543483
0.28	0.513127	-0.233127	0.0543483
0.28	0.513127	-0.233127	0.0543483
0.72	0.513127	0.206873	0.0427963
0.24	0.513127	-0.273127	0.0745985
0.24	0.513127	-0.273127	0.0745985
0.48	0.513127	-0.0331273	0.00109742
1.72	0.513127	1.20687	1.45654
6.52	0.513127	6.00687	36.0825
0.24	0.513127	-0.273127	0.0745985

The mean squared error(MSE): 0.19328250896495996
The Variance for the forecast error: 0.17720386211788544

The average method is not perfroming very bad. The MSE and the variance for the forecast and the prediction is not too high so this model could be considered as the final model.

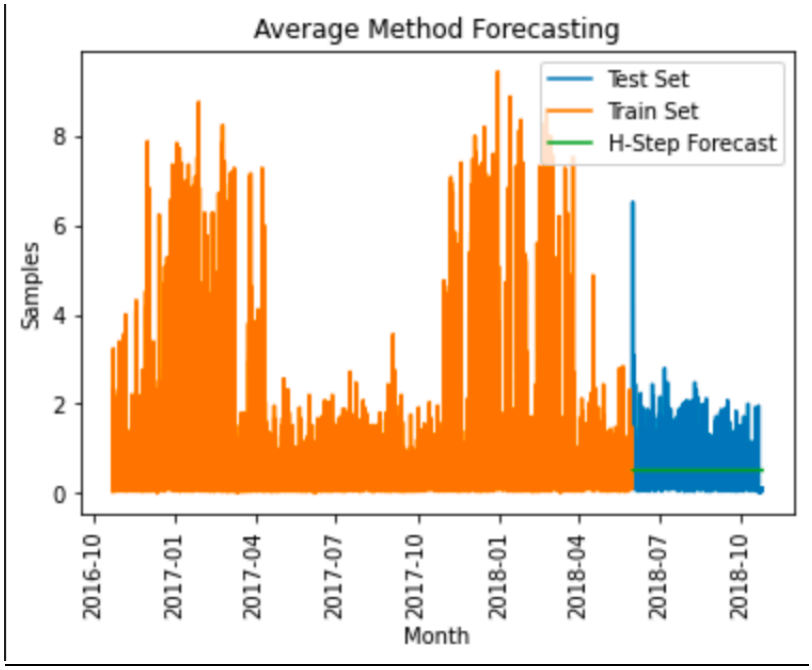

Figure 14 Average method Forecasting

Naïve Method

For train set

Y(t)	Forecast	Errors	Squared Errors
0.04			
0.04	0.04	0	0
0.08	0.04	0.04	0.0016
0.04	0.08	-0.04	0.0016
0.04	0.04	0	0
0.04	0.04	0	0
0.04	0.04	0	0
0.04	0.04	0	0
0.08	0.04	0.04	0.0016

For test set

Y(t)	Forecast	Errors	Squared Errors
0.28	0.84	-1.2	1.44
0.24	0.84	-1.24	1.5376
0.32	0.84	-1.16	1.3456
0.28	0.84	-1.2	1.44
0.28	0.84	-1.2	1.44
0.28	0.84	-1.2	1.44
0.28	0.84	-1.2	1.44
0.72	0.84	-0.76	0.5776
0.24	0.84	-1.24	1.5376
0.24	0.84	-1.24	1.5376
0.48	0.84	-1	1
1.72	0.84	0.24	0.0576
6.52	0.84	5.04	25.4016
0.24	0.84	-1.24	1.5376
0.2	0.84	-1.28	1.6384

The mean squared error(MSE): 0.9571579306424104
The variance for prediction errors: 0.9571579274104329
The mean squared error(MSE): 1.3733274225632282
The Variance for the forecast error: 0.17720386211788544

The naive method is not perfroming very bad. The MSE and the variance for the forecast and the prediction is not too high so this model could be considered as the final model.

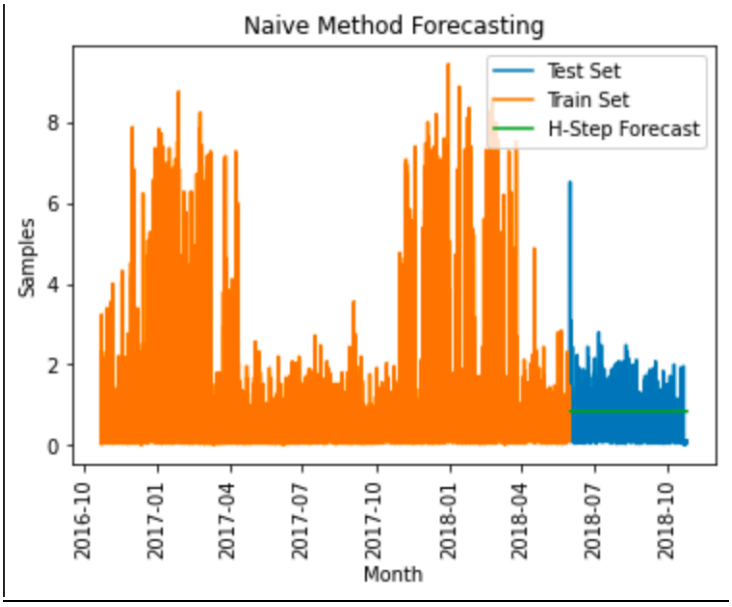

Figure 15 Naive method forecasting

Drift method

For train set

Y(t)	Forecast	Errors	Squared Errors
0.04			
0.04			
0.08	0.04	0.04	0.0016
0.04	0.1	-0.06	0.0036
0.04	0.04	0	0
0.04	0.04	0	0
0.04	0.04	0	0
0.04	0.04	0	0
0.08	0.04	0.04	0.0016
0.04	0.085	-0.045	0.002025
0.08	0.04	0.04	0.0016
0.28	0.084	0.196	0.038416
0.52	0.301818	0.218182	0.0476033
0.04	0.56	-0.52	0.2704
0.24	0.04	0.2	0.04
0.12	0.254286	-0.134286	0.0180327

The mean squared error(MSE): 0.9575999822047048
The variance for prediction errors: 0.9575999451290911

For test set

Y(t)	Forecast	Errors	Squared Errors
0.28	0.840057	-1.2001	1.44025
0.24	0.840114	-1.2401	1.53785
0.32	0.840171	-1.1601	1.34584
0.28	0.840227	-1.2001	1.44025
0.28	0.840284	-1.2001	1.44025
0.28	0.840341	-1.2001	1.44025
0.28	0.840398	-1.2001	1.44025
0.72	0.840455	-0.760102	0.577756
0.24	0.840512	-1.2401	1.53785
0.24	0.840569	-1.2401	1.53785
0.48	0.840625	-1.0001	1.0002
1.72	0.840682	0.239898	0.0575509
6.52	0.840739	5.0399	25.4006
0.24	0.840796	-1.2401	1.53785

The mean squared error(MSE) 1.373551282236607
The Variance for the forecast error: 0.17720386211788544

The Drift method is not perfroming very bad. The MSE and the variance for the forecast and the prediction is not too high so this model could be considered as the final model.

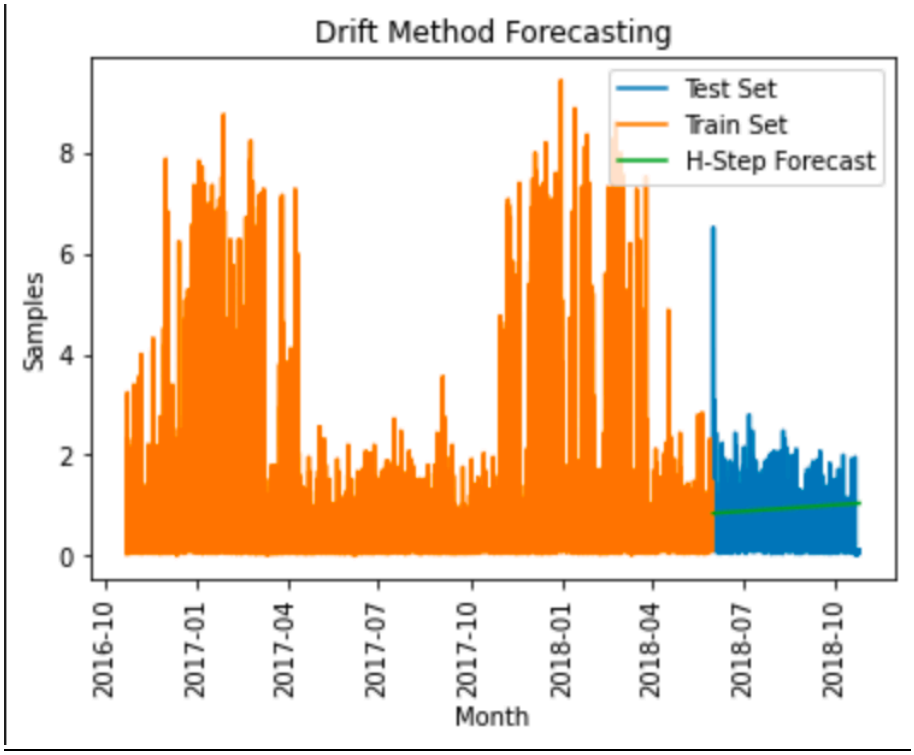

Figure 16 Drift method forecasting

Simple & Exponential Smoothing Method (alpha=0.1)

For train set

Y(t)	Forecast	Errors	Squared Errors
0.04	0.04		
0.04	0.04	-6.93889e-18	4.81482e-35
0.08	0.04	0.04	0.0016
0.04	0.044	-0.004	1.6e-05
0.04	0.0436	-0.0036	1.296e-05
0.04	0.04324	-0.00324	1.04976e-05
0.04	0.042916	-0.002916	8.50306e-06
0.04	0.0426244	-0.0026244	6.88748e-06
0.08	0.042362	0.037638	0.00141662
0.04	0.0461258	-0.00612576	3.7525e-05
0.08	0.0455132	0.0344868	0.00118934
0.28	0.0489619	0.231038	0.0533786
0.52	0.0720657	0.447934	0.200645
0.04	0.116859	-0.0768591	0.00590732

The mean squared error(MSE): 0.8007919635168984
The variance for prediction errors: 0.8007918858190093

For test set

Y(t)	Forecast	Errors	Squared Errors
0.28	0.432248	-0.152248	0.0231794
0.24	0.432248	-0.192248	0.0369592
0.32	0.432248	-0.112248	0.0125996
0.28	0.432248	-0.152248	0.0231794
0.28	0.432248	-0.152248	0.0231794
0.28	0.432248	-0.152248	0.0231794
0.28	0.432248	-0.152248	0.0231794
0.72	0.432248	0.287752	0.0828014
0.24	0.432248	-0.192248	0.0369592
0.24	0.432248	-0.192248	0.0369592
0.48	0.432248	0.0477523	0.00228028
1.72	0.432248	1.28775	1.65831
6.52	0.432248	6.08775	37.0607
0.24	0.432248	-0.192248	0.0369592
0.2	0.432248	-0.232248	0.053939

The mean squared error(MSE): 0.17931269841406125
The Variance for the forecast error: 0.17720386211788544

The SES method is not perfroming very bad. The MSE and the variance for the forecast and the prediction is not too high so this model could be considered as the final model.

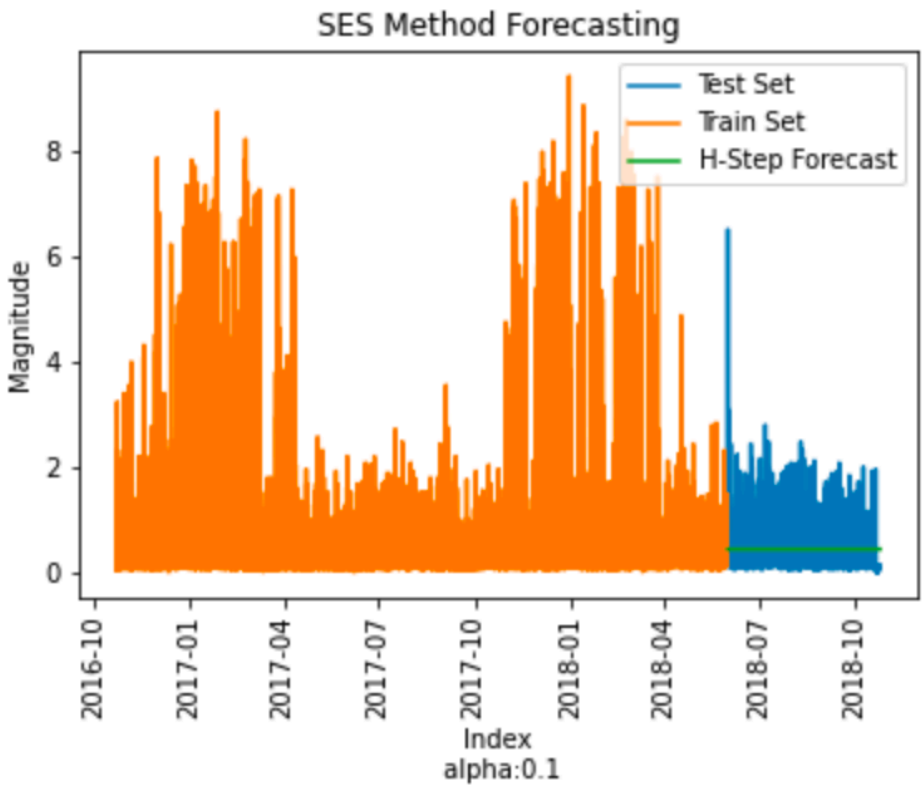

Figure 17 SES method forecasting

8 : Multiple Linear Regression

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  USAGE   R-squared:                       0.985
Model:                            OLS   Adj. R-squared:                  0.985
Method:                 Least Squares   F-statistic:                 4.691e+05
Date:                Tue, 03 May 2022   Prob (F-statistic):               0.00
Time:                        15:31:06   Log-Likelihood:                 10926.
No. Observations:               14073   AIC:                            -2.185e+04
Df Residuals:                   14070   BIC:                            -2.182e+04
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0088      0.001      8.164      0.000       0.007       0.011
COST           4.9135      0.005    967.510      0.000       4.904       4.923
HOLIDAY        0.0302      0.006      5.388      0.000       0.019       0.041
==============================================================================
Omnibus:                    13698.221   Durbin-Watson:                   1.268
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          4533127.117
Skew:                          -4.063   Prob(JB):                         0.00
Kurtosis:                      90.548   Cond. No.                         6.02
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
Singular Values = [14238.18398816   479.67838847   392.33982337]
The conditional number = 6.024154536629359
```

The MLR is perfroming very well. The MSE and the variance for the forecast and the prediction is not too high so this model could be considered as the final model.

```
MSE for prediction errors: 0.012392772440168441
The Variance for prediction error: 0.012392772440168441
The MSE for forecast errors: 0.024100650486289208
The variance for the forecast error: 0.005609868910181114
Q Value for MLR: 5632.96498543066
```

9 : ARMA, ARIMA & SARIMA

From the GPAC table it can be seen that the ARMA model has the order of ARMA(2,0).

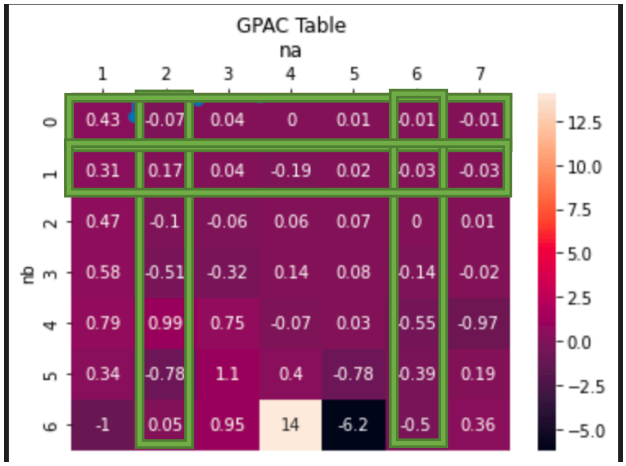

Figure 18 GPAC

From the ACF & PACF after differencing for 12 intervals it is observed that SARIMA$(0,1,1)_{12}$ and ARIMA(0,12,1) are the appropriate models.

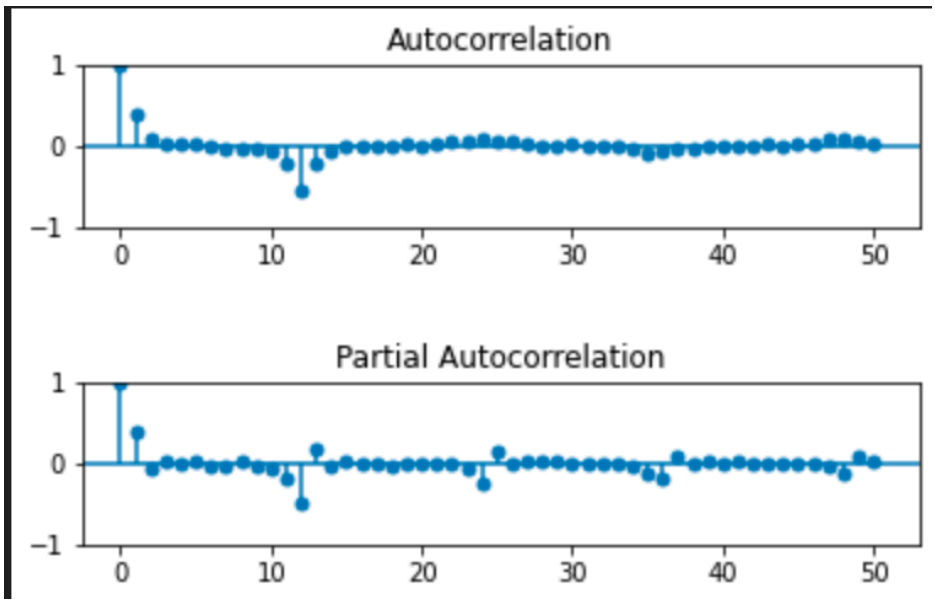

Figure 19 ACF PACF for data after differencing

10 : Levenberg Marquardt algorithm

From the results the AR parameter are -0.5608 and -0.0274 and are well within the confidence intervals and the p-values are also less than 0.05 which mean all the values are not zero and the coefficients are valid.

Since it's an AR process only there's no zero pole cancellations.

As the residual is not white this model is a biased estimator.

ARMA(2,0)

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                  USAGE   No. Observations:                17592
Model:                 ARIMA(2, 0, 0)   Log Likelihood              -20919.859
Date:                Wed, 04 May 2022   AIC                          41845.718
Time:                        02:00:51   BIC                          41869.044
Sample:                  10-22-2016   HQIC                         41853.397
                       - 10-24-2018
Covariance Type:                 opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.5608      0.003    190.080      0.000       0.555       0.567
ar.L2          0.0274      0.005      5.908      0.000       0.018       0.037
sigma2         0.6316      0.002    362.488      0.000       0.628       0.635
===================================================================================
Ljung-Box (L1) (Q):                  91.12   Jarque-Bera (JB):            646174.65
Prob(Q):                              0.00   Prob(JB):                         0.00
Heteroskedasticity (H):               0.35   Skew:                             3.62
Prob(H) (two-sided):                  0.00   Kurtosis:                        31.79
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
The residual is NOT white
```

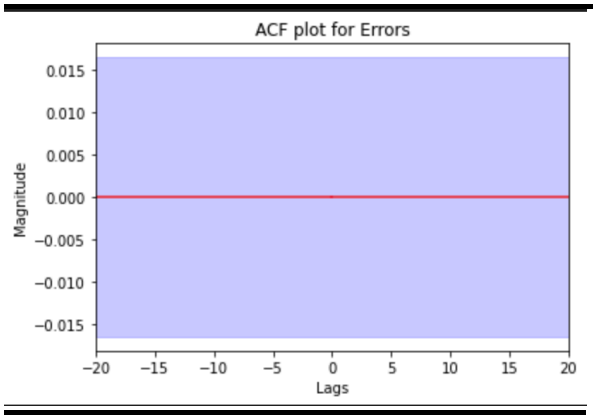

Figure 20 ACF for residuals (ARMA)

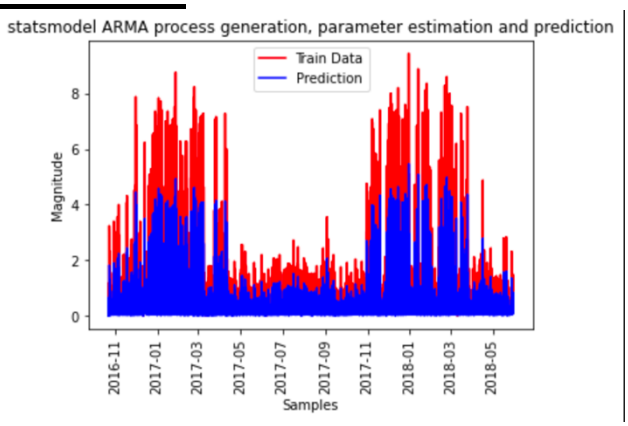

Figure 21 Train and prediction for ARMA(2,0)

```
MSE for prediction errors: 0.7493141188289068
The Variance for prediction error: 0.7046485763119538
MSE for forecast errors: 0.1610694147743025
The Variance for forecast error: 0.13580866005762074
```

ARIMA(0,12,1)

```
                               SARIMAX Results
==========================================================================================
Dep. Variable:                           USAGE   No. Observations:                 14073
Model:                 ARIMA(0, 12, 1)   Log Likelihood             -105970.031
Date:                  Wed, 04 May 2022   AIC                         211944.062
Time:                          03:07:10   BIC                         211959.165
Sample:                      10-22-2016   HQIC                        211949.089
                           - 05-31-2018
Covariance Type:                    opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ma.L1         -1.0000      0.061    -16.273      0.000      -1.120      -0.880
sigma2      2.059e+05   1.27e+04     16.252      0.000    1.81e+05    2.31e+05
===================================================================================
Ljung-Box (L1) (Q):              11422.77   Jarque-Bera (JB):            132554.37
Prob(Q):                             0.00   Prob(JB):                         0.00
Heteroskedasticity (H):              0.84   Skew:                             0.00
Prob(H) (two-sided):                 0.00   Kurtosis:                        18.04
===================================================================================
```

```
MSE for Prediction: 205673.5843863718
Variance for Prediction: 205673.58438051643
MSE for forecast: 4.3627046884603345e+59
Variance for Forecast: 3.605782776411029e+59
```

From the plot below we can see that the residuals are not white and hence this is a biased estimator. The MSE for the prediction and the forecast is also very high which indicated the model is not performing very well.
Hence ARIMA(0.12,1) cannot be used for forecasting the electricity usage.

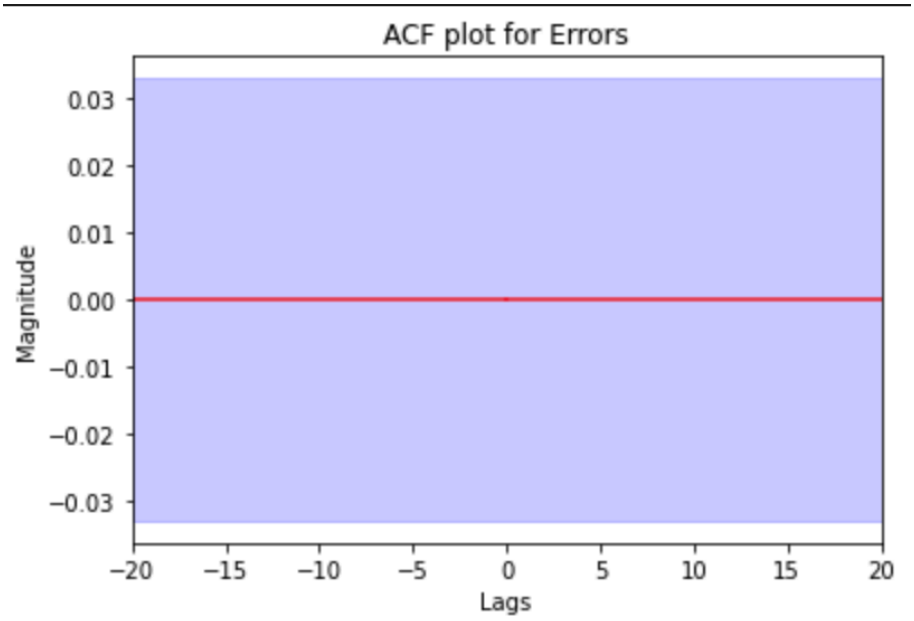

Figure 22 ACF for ARIMA

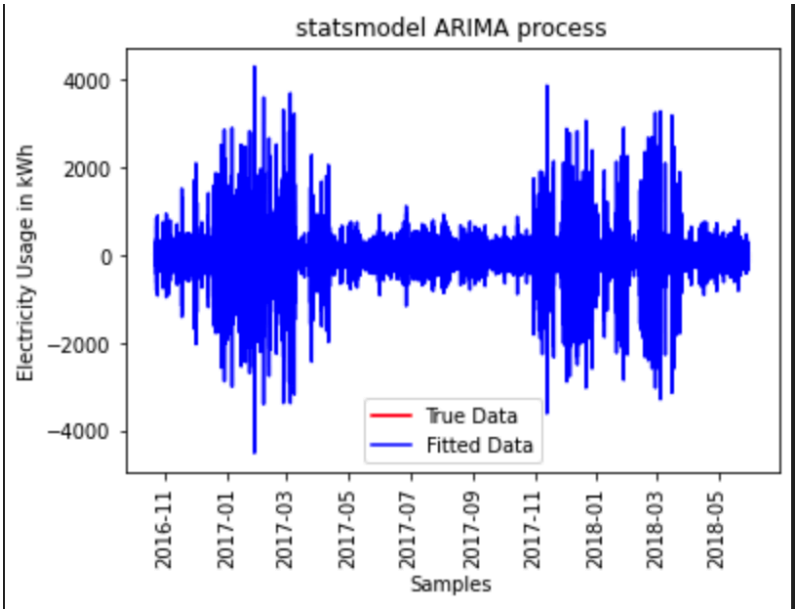

Figure 23 ARIMA Train vs Prediction.

SARIMA

```
                                SARIMAX Results
==========================================================================================
Dep. Variable:                           USAGE   No. Observations:                14073
Model:             SARIMAX(0, 1, [1], 12)   Log Likelihood              -18035.779
Date:                    Wed, 04 May 2022   AIC                           36075.558
Time:                            03:17:00   BIC                           36090.660
Sample:                        10-22-2016   HQIC                          36080.584
                             - 05-31-2018
Covariance Type:                     opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ma.S.L12      -0.9512      0.001   -635.920      0.000      -0.954      -0.948
sigma2         0.7599      0.003    297.113      0.000       0.755       0.765
===================================================================================
Ljung-Box (L1) (Q):                2084.18   Jarque-Bera (JB):            369093.01
Prob(Q):                              0.00   Prob(JB):                         0.00
Heteroskedasticity (H):               1.01   Skew:                             4.07
Prob(H) (two-sided):                  0.84   Kurtosis:                        26.74
===================================================================================
```

```
MSE for SARIMA model (prediction): 0.760472481601338
Variance for SARIMA model (prediction): 0.7604720293872358
```

```
MSE for SARIMA model (forecast): 0.16843425785116464
Variance for SARIMA model (forecast): 0.16621819723158113
```

The SARIMA model is not perfroming very bad. The MSE and the variance for the forecast and the prediction is not too high so this model could be considered as the final model.

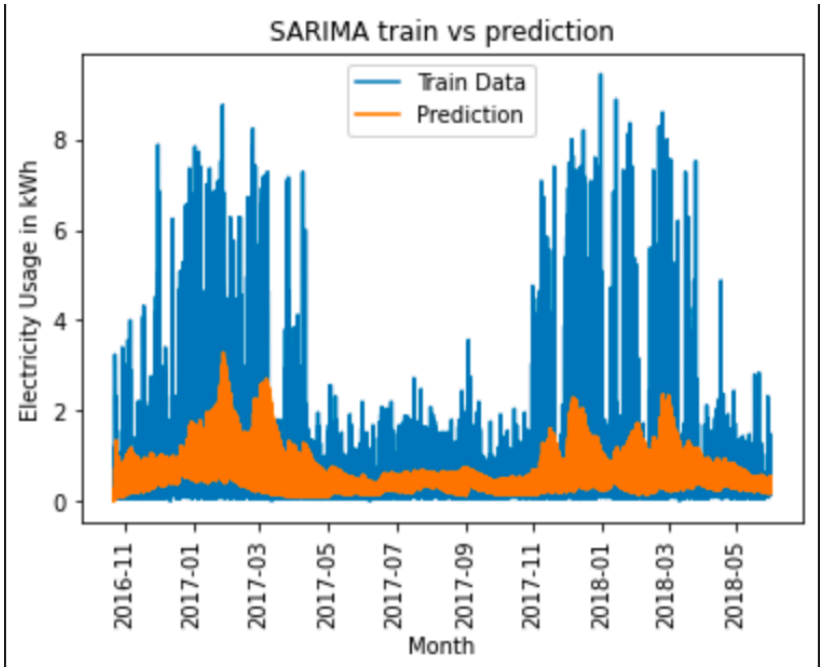

Figure 24 SARIMA train vs prediction

Table 1

Model	MSE Prediction	MSE Forecast	Variance Prediction	Variance Forecast
Average	0.8391	0.1932	0.8372	0.1772
Naïve	0.9571	0.1373	0.9571	0.1772
Drift	0.9575	1.3735	0.9575	0.1772
Holt Winters	0.7407	0.1608	0.7407	0.1608
SES	0.8007	0.1793	0.8007	0.1772
MLR	0.0123	0.0241	0.0123	0.0056
ARMA	0.7493	0.1610	0.7046	0.1358
ARIMA	205673.58	4.3627×10^{59}	205673.58	3.6057
SARIMA	0.7604	0.1684	0.7604	0.1662

11 : Best Model Selection

Form table 1 it can be seen that multiple linear regression has the least mean squared error (MSE) and variance for prediction and the forecast. Hence the best model for this data is the multiple linear regression model (MLR).

12 : h-step prediction

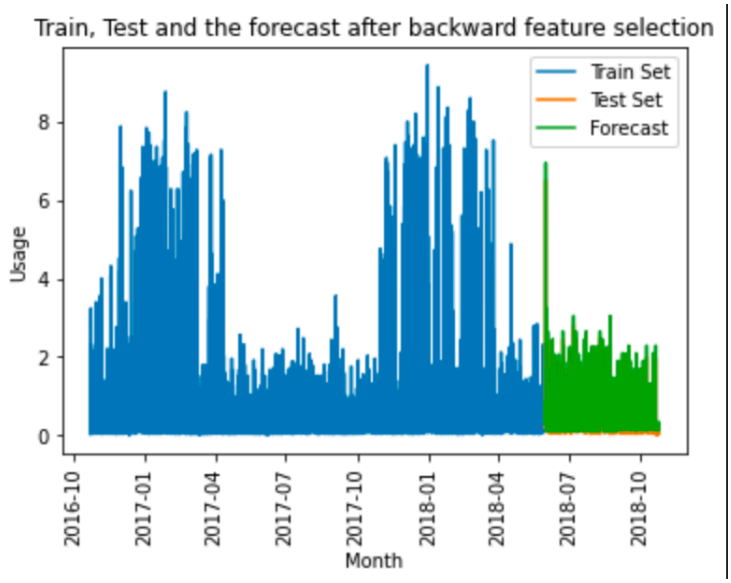

Figure 25 h-step prediction for MLR

From the above figure it can be seen that the forecast is exactly in line with the test set which means the model is forecasting the values very well.

13 : Conclusion

The multiple regression model has given the best results for prediction as well as the forecast. It had the least MSE than any other model used for prediction. The plot also shows that the forecasts are exactly in line with the test set.
A better model than this could be the LSTM model which uses deep learning and could've given better results.

14: References

Data: https://www.kaggle.com/jaganadhg/house-hold-energy-data

15 : Appendix

```
#%%
from tracemalloc import start
from scipy.stats import chi2
from cProfile import label
from multiprocessing.spawn import import_main_path
from unicodedata import category, name
from importlib_metadata import distribution
import pandas as pd
# from Lab4 import X_test, X_train
import numpy as np
from pyparsing import col
import toolbox as tx
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime
from sklearn.model_selection import train_test_split
from pandas.tseries.holiday import USFederalHolidayCalendar as calendar
import math
from statsmodels.tsa.seasonal import STL
import statsmodels.tsa.holtwinters as ets
import statsmodels.api as sm
from scipy import signal
# %%
def difference(dataset, interval =1):
    diff = []
    for i in range(interval, len(dataset)):
        value = dataset[i]- dataset[i-interval]
        diff.append(value)
    return diff

def log_trans(column):
    lg= []
    for i in range(len(column)):
        value = math.log(column[i])
        lg.append(value)
    return lg
#%%
#%%
df  =
pd.read_csv("/Users/atharvah/GWU/Untitled/personal/Time_Series/Labs/D202.csv")
df.DATE = pd.to_datetime(df.DATE + ' ' + df["END TIME"])
df = df.drop(columns=["START TIME","END TIME"])
df = df.set_index(df.DATE)
# 1: spring,2: summer,3: fall,4: winter
df["season"]= df.index.month%12 // 3 + 1
#%%
# Citation: JAGANADH GOPINADHAN
# Check if holiday or not
```

```
cal = calendar()
holidays = cal.holidays(start = df.index.min(), end = df.index.max())
df["HOLIDAY"] = df.index.strftime('%Y-%m-%d').isin(holidays.strftime('%Y-%m-%d'))
df["HOLIDAY"] = pd.Categorical(df["HOLIDAY"])
df.season = pd.Categorical(df.season)
#%%
df.COST = df.COST.str[1:]
df.COST = df.COST.astype(float)
#%%

#%%
# Data Pre-processing
display(df.head())
display(df.tail())
missing_vals = df.isna().sum()
print("The numeber of Missing Values:", missing_vals)


#%%
# Dropping column with empty value and irrelevant values
df = df.drop(columns=["NOTES","TYPE","UNITS"])


# Displaying the Shape of the dataframe
print(f"There are {len(df.index)} rows and {len(df.columns)} columns in the
dataset.")
display(df.head())

#%%
# Citation : SANKET SHARMA
# Hourly
hourly_usage_df = df.resample("1H", on="DATE").sum()
display(hourly_usage_df.head())

# Daily
daily_usage_df = hourly_usage_df.resample("1D").sum()
display(daily_usage_df.head())

# Monthly
monthly_usage_df = daily_usage_df.resample("1M").sum()
display(monthly_usage_df.head())

#%%
monthly_usage_df["per_unit_cost"] =
monthly_usage_df["COST"]/monthly_usage_df["USAGE"]
monthly_usage_df["year"] = monthly_usage_df.index.year.astype(str)
monthly_usage_df["month_name"] = monthly_usage_df.index.month_name().str[:3]

monthly_usage_df["usage_month"] = monthly_usage_df["month_name"] + "-" +
monthly_usage_df["year"]
```

```
monthly_usage_df = monthly_usage_df.drop(columns=["year", "month_name"],
errors="ignore")

hourly_usage_df["year"] = hourly_usage_df.index.year.astype(str)
hourly_usage_df["month_name"] = hourly_usage_df.index.month_name().str[:3]

hourly_usage_df["usage_month"] = hourly_usage_df["month_name"] + "-" +
hourly_usage_df["year"]
hourly_usage_df["season"]= hourly_usage_df.index.month%12 // 3 + 1
hourly_usage_df.season = pd.Categorical(hourly_usage_df.season)

hourly_usage_df = hourly_usage_df.drop(columns=["year", "month_name"],
errors="ignore")
hourly_usage_df["HOLIDAY"] = hourly_usage_df.index.strftime('%Y-%m-
%d').isin(holidays.strftime('%Y-%m-%d')).astype(int)
hourly_usage_df["HOLIDAY"] = pd.Categorical(hourly_usage_df["HOLIDAY"])

display(hourly_usage_df.head())
#%%
tx.ACF_PACF_Plot(hourly_usage_df.USAGE,25)
#%%
# Target variable vs Time
plt.plot(df.USAGE)
plt.title("Usage in every 15 Minutes")
plt.xlabel("Month")
plt.ylabel("Electricity Usage in KWh")
plt.xticks(rotation = 90)
plt.show()
#%%
# Change in unit rate montly
plt.plot(monthly_usage_df.index,monthly_usage_df.per_unit_cost)
plt.title("Monthly Change in Price per unit")
plt.xlabel("Month")
plt.ylabel("Electricity price/unit ($)")
plt.xticks(rotation = 90)
plt.show()
#%%
# Hourly change in usage of electricity
plt.plot(hourly_usage_df.USAGE)
plt.title("Hourly Usage")
plt.xlabel("Month")
plt.ylabel("Electricity Usage in KWh")
plt.xticks(rotation = 90)
plt.show()
#%%

#%%
datatypes = hourly_usage_df.dtypes
display(datatypes)
#%%
X_train, X_test = train_test_split(hourly_usage_df,test_size=0.2,shuffle=False)
```

```
# %%
# tx.Cal_rolling_mean_var(hourly_usage_df.USAGE)
# Since the rolling mean and the rolling variance turns into a straight line the
data is stationary.
# %%
tx.ADF_Cal(hourly_usage_df.USAGE)
# We reject the null hypothesis hence the data is staitonary according to ADF test
# %%
tx.kpss_test(hourly_usage_df.USAGE)
# We failt reject the null hypothesis hence the data is staitonary according to
kpss test
#%%
# %%
tx.stem_plot(hourly_usage_df.USAGE,20,name="Hourly electricity usage")
# %%
sns.pairplot(hourly_usage_df,kind="kde").set(title='Hourly Usage')
# %%
df1 = hourly_usage_df.corr()
sns.heatmap(df1,annot=True).set(title = "Correlation matrix for hourly Usage ")
# %%
STL = STL(hourly_usage_df.USAGE)
res = STL.fit()
fig = res.plot()
plt.show()
#%%
S=  res.seasonal
T =  res.trend
R = res.resid
#%%
plt.plot(R,label="Residual")
plt.plot(S,label="Seasonal")
plt.plot(T,label="Trend")
plt.legend(loc = "best")
plt.xlabel("months")
plt.ylabel("Electricity Usage in kWh")
plt.title("Plot for seasonality, trend and residual")
plt.xticks(rotation = 90)
plt.show()
# %%
adj_seasonal  = hourly_usage_df.USAGE-S
plt.plot(hourly_usage_df.USAGE,label="original")
plt.plot(adj_seasonal,label="seasonally adjusted")
plt.xlabel("Month")
plt.ylabel("Electricity Usage in kWh")
plt.title("Orginal data vs seasonally adjusted data")
plt.legend(loc = "best")
plt.xticks(rotation = 90)
plt.show
# %%
F = np.maximum(0,1-np.var(np.array(R))/np.var(np.array(S)+np.array(R)))
print(f"The strength of seasonality for this data set is:{F}")
```

```
F1 = np.maximum(0,1-np.var(np.array(R))/np.var(np.array(T)+np.array(R)))
print(f"The strength of trend for this data set is:{F1}")
#%%
detrended_pass= hourly_usage_df.USAGE-T
plt.plot(hourly_usage_df.USAGE,label="original")
plt.plot(detrended_pass,label="Detrended data")
plt.xlabel("Month")
plt.ylabel("Electricity Usage in kWh")
plt.title("Orginal data vs detrended data")
plt.legend(loc = "best")
plt.xticks(rotation = 90)
plt.show
# %%
holtt = ets.ExponentialSmoothing(X_train.USAGE,seasonal ="add",damped_trend =
False).fit()
holtf = holtt.forecast(steps=len(X_test.USAGE))
holtf = pd.DataFrame(holtf).set_index(X_test.index)

#%%
holtt.summary()
#%%
error_hw = X_test.USAGE - holtf[0]
er_sqd_hw = error_hw**2
pred_error_hw = X_train.USAGE-holtt.fittedvalues

rk= tx.stem_plot(pred_error_hw,5,name="predcition errors")
qv = (len(X_train))*(np.array(rk[1:])**2)
af = print(f"Name of test:Holt's Winter Method,\n 'Q value':{qv},MSE for prediction
errors: {np.mean((X_train.USAGE-holtt.fittedvalues)**2)},MSE for forecast
errors:{np.mean(er_sqd_hw)},Variance of prediction errors:{np.var(X_train.USAGE-
holtt.fittedvalues)},Variance of forecast errors: {np.var(error_hw)}, Correlation
Coeff:{np.corrcoef(er_sqd_hw,X_test.USAGE)}")
# tab = tab.append(af,ignore_index = True)

fig, ax = plt.subplots()
ax.plot(X_train.USAGE,label = "Training set")
ax.plot(X_test.USAGE,label ="Test")
ax.plot(holtf[0],label = "Holt's Winter Method")
ax.set_xlabel("Months")
ax.set_ylabel("Electricity Usage in kWh")
ax.set_title("Holt's Winter Method")
plt.xticks(rotation = 90)
ax.legend()
plt.show()
# %%
# Average forecasting
tx.average_forecasting(hourly_usage_df.USAGE)
# %%
# Naive forecasting
tx.N_forecasting(hourly_usage_df.USAGE)
# %%
```

```
# SES forecasting
tx.ses_forecasting(hourly_usage_df.USAGE,alpha=0.1)
# %%
# Drift method forecasting
tx.d_forecasting(hourly_usage_df.USAGE)
# %%
X= hourly_usage_df.loc[:,hourly_usage_df.columns!='USAGE']
X= X.drop(columns="usage_month")
Y = hourly_usage_df.loc[:,hourly_usage_df.columns=='USAGE']
X = sm.add_constant(X)
#%%
X_tr , X_ts,Y_tr, Y_ts = train_test_split(X,Y,test_size=0.2, shuffle=False)
#%%
Xm = np.array(X_tr)
Ym = np.array(Y_tr)
h = np.matmul(Xm.T,Xm)
# %%
s, d, v = np.linalg.svd(h)
# %%
print("Singular Values =",d)
print("The conditional number =",np.linalg.cond(Xm))
# %%
# Coefficients from MLR
model = sm.OLS(Y_tr,X_tr).fit()
print(model.summary())
# %%
X_tr = X_tr.drop(columns="season")
# %%
model = sm.OLS(Y_tr,X_tr).fit()
print(model.summary())
Xm = np.array(X_tr)
Ym = np.array(Y_tr)
h = np.matmul(Xm.T,Xm)
s, d, v = np.linalg.svd(h)
print("Singular Values =",d)
print("The conditional number =",np.linalg.cond(Xm))
# %%
X_ts = X_ts.drop(columns="HOLIDAY")
prediction = model.predict(X_ts)
print(prediction)
# %%
plt.plot(Y_tr,label = "Train Set")
plt.plot(Y_ts,label = "Test Set")
plt.plot(prediction, label = "Forecast")
plt.legend()
plt.xlabel("Month")
plt.ylabel("Usage")
plt.title("Train, Test and the forecast after backward feature selection")
plt.xticks(rotation = 90)
plt.show()
# %%
```

```
pred_error = Y_tr.subtract(model.fittedvalues,axis=0)
print(pred_error)

pred_error = np.array(pred_error)
# %%
rk = tx.stem_plot(pred_error,lag = 50,name = "Prediction Error")
print("ACF of resioduals:",rk)
# %%
# var_pred = np.sqrt((1/(len(pred_error)-len(X_test.columns)))*((sum(i*i for i in
pred_error))))
print("MSE for prediction errors:",np.mean(np.square(pred_error)))
print("The Variance for prediction error:",np.var(pred_error))

for_error = Y_ts.USAGE- prediction
print("The MSE for forecast errors:",np.mean(np.square(for_error)))
print("The variance for the forecast error:",np.var(for_error))

Q = len(X_train) * np.sum(np.square(rk[20:]))
sm.stats.acorr_ljungbox(for_error, lags=[20], return_df=True)
print("Q Value for MLR:",Q)


# %%
tx.arma_dat(hourly_usage_df.USAGE,na =7,nb = 7)
# %%
model = sm.tsa.ARIMA(hourly_usage_df.USAGE, order=(2, 0, 0), trend="n", ).fit()
print(model.summary())
# Prediction
model_hat = model.predict(start=0, end=len(X_train) - 1)
pred_error = X_train.USAGE[1:] - model_hat[:-1]
re = tx.stem_plot(pred_error, 20, name="Errors")
Q = len(X_train) * np.sum(np.square(re[20:]))
DOF = 20 - 2 - 0
alfa = 0.01
chi_critical = chi2.ppf(1 - alfa, DOF)
if Q < chi_critical:
    print("The residual is white")
else:
    print("The residual is NOT white")
lbvalue, pvalue = sm.stats.acorr_ljungbox(pred_error, lags=[20], return_df=True)
print(lbvalue)
print(pvalue)
plt.figure()
plt.plot(X_train.USAGE, "r", label="Train Data")
plt.plot(model_hat, "b", label="Prediction")
plt.xlabel("Samples")
plt.ylabel("Magnitude")
plt.legend()
plt.xticks(rotation = 90)
plt.title("statsmodel ARMA process generation, parameter estimation and
prediction")
```

```
plt.show()
#%%
forecast = model.predict(start=len(X_train), end=len(hourly_usage_df))
for_error = X_test.USAGE-forecast
re = tx.stem_plot(for_error, 20, name="Errors")
Q = len(X_test) * np.sum(np.square(re[20:]))
DOF = 20 - 2 - 0
alfa = 0.01
chi_critical = chi2.ppf(1 - alfa, DOF)
if Q < chi_critical:
    print("The residual is white")
else:
    print("The residual is NOT white")
lbvalue, pvalue = sm.stats.acorr_ljungbox(pred_error, lags=[20], return_df=True)
print(lbvalue)
print(pvalue)
plt.figure()
plt.plot(X_test.USAGE, "r", label="Train Data")
plt.plot(forecast, "b", label="Forecast")
plt.xlabel("Samples")
plt.ylabel("Magnitude")
plt.legend()
plt.title("statsmodel ARMA process generation, parameter estimation and
prediction")
plt.show()
#%%
print("MSE for prediction errors:",np.mean(np.square(pred_error)))
print("The Variance for prediction error:",np.var(pred_error))

print("MSE for forecast errors:",np.mean(np.square(for_error)))
print("The Variance for forecast error:",np.var(for_error))
#%%
diff1 = tx.difference(hourly_usage_df.USAGE,12)
tx.ACF_PACF_Plot(diff1,50)
# %%
# ARIMA(0,12,1)
model = sm.tsa.ARIMA(X_train.USAGE, order=(0, 12, 1) ).fit()
print(model.summary())
# Prediction
model_hat = model.predict(start=0, end=len(X_train))
e =  X_train.USAGE - model_hat

print("MSE for Prediction:",np.mean(np.square(e)))
print("Variance for Prediction:",np.var(e))
# forecast

forecast = model.predict(start = len(X_train),end  = len(hourly_usage_df))
e =  X_test.USAGE - forecast

print("MSE for forecast:",np.mean(np.square(e)))
print("Variance for Forecast:",np.var(e))
```

```
re = tx.stem_plot(e, 20, name="Errors")
Q = len(X_train.USAGE) * np.sum(np.square(re[20:]))
DOF = 20 - 0 - 1
alfa = 0.01
chi_critical = chi2.ppf(1 - alfa, DOF)
if Q < chi_critical:
    print("The residual is white")
else:
    print("The residual is NOT white")
lbvalue, pvalue = sm.stats.acorr_ljungbox(e, lags=[20], return_df=True)
print(lbvalue)
print(pvalue)
plt.figure()
plt.plot(X_train.USAGE, "r", label="True Data")
plt.plot(model_hat, "b", label="Fitted Data")
plt.xlabel("Samples")
plt.ylabel("Electricity Usage in kWh")
plt.legend()
plt.xticks(rotation = 90)
plt.title("statsmodel ARIMA process")
plt.show()
#%%
plt.plot(X_test.USAGE,label = "Test Data")
plt.plot(forecast,label = "Forecast")
plt.legend()
plt.xlabel("Months")
plt.ylabel("Electricity Usage in kWh")
plt.title("statsmodel ARIMA process")
plt.xticks(rotation = 90)
plt.show()
#%%
# SARIMA(0,1,1)
model=sm.tsa.statespace.SARIMAX(X_train.USAGE,order=(0, 0,
0),seasonal_order=(0,1,1,12))
results=model.fit()

print(results.summary())

prediction = results.predict(start = 0,end =len(X_train.USAGE),dynamic = False)
plt.plot(X_train.USAGE,label = "Train Data")
plt.plot(prediction,label ="Prediction")
plt.legend()
plt.xlabel("Month")
plt.ylabel("Electricity Usage in kWh")
plt.title("SARIMA train vs prediction")
plt.xticks(rotation = 90)
#%%
prediction_error = X_train.USAGE-prediction
print("MSE for SARIMA model (prediction):",np.mean(np.square(prediction_error)))
print("Variance for SARIMA model (prediction):",np.var(prediction_error))
```

```
# %%
forecast = results.predict(start = len(X_train),end
=len(hourly_usage_df.USAGE),dynamic = False)
plt.plot(X_test.USAGE,label = "Test Data")
plt.plot(forecast,label ="Prediction")
plt.legend()
plt.xlabel("Month")
plt.ylabel("Electricity Usage in kWh")
plt.title("SARIMA train vs prediction")
plt.xticks(rotation = 90)
# %%
forecast_error = X_test.USAGE-forecast
print("MSE for SARIMA model (forecast):",np.mean(np.square(forecast_error)))
print("Variance for SARIMA model (forecast):",np.var(forecast_error))
# %%
re = tx.stem_plot(forecast_error, 20, name="Errors")
Q = len(X_train.USAGE) * np.sum(np.square(re[20:]))
DOF = 20 - 0 - 1
alfa = 0.01
chi_critical = chi2.ppf(1 - alfa, DOF)
if Q < chi_critical:
    print("The residual is white")
else:
    print("The residual is NOT white")
lbvalue, pvalue = sm.stats.acorr_ljungbox(e, lags=[20], return_df=True)
print(lbvalue)
print(pvalue)
# %%
```